RNAseq_QC

Matthew Taliaferro

6/14/2020

Overview

In this document, we will examine RNAseq data collected over a timecourse of differentiation from mouse embryonic stem cells to cortical glutamatergic neurons (Hubbard et al, F1000 Research (2013)). In this publication, the authors differentiated mESCs to neurons using a series of *in vitro* culture steps over a period of 37 days. During this timecourse, samples were extracted at selected intervals for transcriptome analysis. Importantly, for each timepoint, either 3 or 4 samples were taken for RNA extraction, library preparation and sequencing. This allows us to efficiently use the statistical frameworks provided by the DESeq2 package to identify genes whose RNA expression changes across the timecourse.

Cells were grown in generic differentiation-promoting media (LIF^-) for 8 days until aggreates were dissociated and replated in neuronal differentiation media. This day of replating was designated as *in vitro* day 0 (DIV0). The timepoints taken before this replating therefore happened at "negative" times (DIV-8 and DIV-4). Because naming files with dashes or minus signs can cause problems, these samples are referred to as DIVminus8 and DIVminus4. Following the replating, samples were taken at days 1, 7, 16, 21, and 28 (DIV1, DIV7, DIV16, DIV21, and DIV28).

Today we will focus on some Quality Control steps that are good ideas to do for every RNAseq dataset you encounter, whether produced by yourself or someone else.

Quantification of reads with salmon

We recently learned about the RNAseq quantification tool salmon. We won't rehash the details here about how salmon works. For our purposes, we just need to know that salmon reads in a fastq file of sequencing reads and a fasta file of transcript sequences to be quantified. Let's take a look at this fasta file of transcripts:

```
>ENSMUST00000096114
AACCAAGCTGCCCGGTGGAGGCCTGGGCAGAAAGTCGCCGGGGCCTCAGTCACCCCTGTGGGTTCTCCGCGCTCTCAGGAGCAGACGTCTCGCGCAGGGCACCGCCCTGCCTGACCCACG
CGGGGACACAGTGGTAGAGTGGGACGGCGAGGCTCCGCGTCTCTGTCAGCGGCCTAGCCGAAGGGCAGCCGCCATGTCCCGGGTACTGGTCGTGGGCGCTGGGCTAACCGGAAGTCTGTG
TGCCGCGCTGCTGAGGAAGGAAATAACCGCCCCCCTGTACCTCGGCCTGTGAGACAAGGGTGGGGATATAGGGGGAAGAATGATTACTGCCAGCAGTCCTCATAATCCCCGATGCACAGC
TGACTTGGGAGCTCAGTACATCACCTGCTCTCCTCATTATGTCAAAGAGCACCAAAATTTTTATGAGGAACTGTTAGCTCATGGAATTTTGAAGCCTCTGACATCCCCCATTGAAGGAAT
GAAAGGGAAGGAAGGAGATTGCAACTTTGTGGCACCTCAAGGATTTTCTTCAGTTATCAAGTACTACTTGAAAAAGTCAGGTGCAGAAGTCTCCCTCAAGCACTGTGTGACTCAGATCCA
CCTGAAAGATAACAAGTGGGAAGTCTCCACAGACACTGGCTCTGCTGAGCAGTTTGACCTTGTCATCCTCACCATGCCAGCTCCTCAGATTCTGGAACTTCAAGGTGACATTGTGAACTT
AATTAGTGAACGCCAGAGGGAGCAACTGAAATCTGTGAGCTACTCCTCTCGCTATGCTCTGGGCCTCTTTTATGAAGTAGGCATGAAGATTGGTGTCCCTTGGTCCTGCCGCTACCTCAG
CAGTCACCCCTGCATATGCTTCATCTCCATTGATAATAAGAAGCGCAATATAGAGTCATCAGAATGTGGTCCATCCGTGGTGATCCAAACCACTGTCCCATTTGGAGTTCAACACTTGGA
GGCCAGTGAGGCGGATGTGCAGAAGTTAATGATCCAGCAATTGGAAACCATTCTGCCGGGTTTGCCTCAGCCAGTTGCTACCATATGCCATAAATGGACATATTCACAGGTTACAAGCTC
AGTTTCCGACAGACCTGGTCAGATGACTCTTCATCTCAAGCCTTTCCTGGTGTGCGGAGGGGATGGATTTACTCACTCCAACTTCAATGGCTGCATCTCCTCTGCCCTGAGTGTCATGAA
AGTTTTAAAGCGTTATATTTAGTGTCTGTGTTCTTGTTCTCTACATTTACTTTAGGATTTTTGTTTTCACAATCCTCTGTTATTAATTGTTCTATTTTCCTTTCTGTGAAGAGAAATTAC
TTTGGAAAATGTCTTCACTTACTGTTATTTTTAATATGAAATATAAAAACAATGTTATAATTTTGATAGCTACCACCCATAGTAAAGTGATAACTTCTTAGGCTATATTTCTTCCTCACT
TTTCTGACTTTATTACTAATCCTTATTGGAGAAAAATGGTGCTTCTAAGTCACAAATAACCCAAGCCCAGAATTATACTAAACCAAAGTTCTCATTCATGGAAAATCTTTATTATCCTTT
TGAAAACTTCAGTTCTAAATAAATAACAGTTACTACC
>ENSMUST00000174016
CGGACGGAGCAGAACGGAAGCAGAGGGACAAAGTAGCGGATTCCTGACTTCTGGCAAGGGCCCCTTCCTGGGTTTCACGCCGCTGCGTTCTGGTCGAGGCTGGAACGCGCGCCCAGCCCT
TTGTGTGGCTGTGGTCCGAGGCTACGGCTCAGAAGCCGAGCGCTGGGAACGGCAGGACTGCCGAGCCTCAAGTAGGGCTGTCGAGCGCGGGCTCCTGCACCAGGAAAGAACAATCAGAGC
CCAGGCTGTCCTAGAACTCCCGGAGATCCTGTCTCAGCCTTCCTAGTGTTTAGGTAATCGTCTAGGTCACCACGCATCTTTATGCTATTGTTAAGACTTGTTGATAAAGGGGTCTTTGCT
GAGAATGGTTAAAGACAGGTCCTGGAAGACACTAGCATAGAACTGTATGGACAGGTCCTGGAAGACACTGTTTGGAATGTTTTTTTTTTTTTTTTTTTCCAACGGAAGAATAATGAGA
GGTGAGACAGACAGAAGGAAGGAAGAAAATCAACCTCAGAATGCAGCTTTCCTGGCAACTTTTTTGGGGATCAGAGCTAGCGATTCCTTGAATGCCTTGTTCCTTGCTGG
```

Looks like we have ensembl transcript IDs, which is a good idea. I can tell because they start with 'ENS'. Using ensembl IDs as transcript names will allow us to later collate transcript expression levels into gene expression levels using a database that relates transcripts and genes. More on that later.

Making a transcriptome index

The first step in quantifying these transcripts is to make an index from them. This is done as follows:

```
salmon index -t <transcripts.fa> -i <transcripts.idx> --type quasi -k <k>
```

Here, transcripts.fa is a path to our fasta file, transcripts.idx is the name of the index that will be created, and k is the length of the kmers that will be used in the hash table related kmers and transcripts. k is the length of the minimum accepted match for a kmer in a read and a kmer in a transcript. Longer kmers (higher values of k) will therefore be more stringent, and lowering k may improve mapping sensitivity at the cost of some specificity. You may also see here how read lengths can influence what value for k you should choose.

Consider an experiment where we had 25 nt reads (this was true wayyyyyy back in the old, dark days of high-throughput sequencing). What's going to happen if I quantify these reads using an index where the kmer size was set to 29? Well, nothing will align. The index has represented the transcriptome in 29 nt chunks. However, no read will match to these 29mers because there are no 29mers in these reads! *As a general rule of thumb, for reads 75 nt and longer (which is the bulk of the data produced nowadays), a good value for k that maximizes both specificity and sensitivity is 31.* However, datasets that you may retrieve from the internet, particularly older ones, may have shorter read lengths, so keep this is mind when defining k.

Quantifying reads against your index

Once we have our index, we can quantify transcripts in the index using reads from our fastq files.

```
salmon quant --libType A -p 8 --seqBias --gcBias --validateMappings -1 <forwardreads.fastq> -2 <reverse
```

In this command, our forward and reverse read fastq files are supplied to -1 and -2, respectively. If the experiment produced single end reads, -2 is omitted. <transcripts.idx> is the path to the index produced in the previous step. I'm not going to go through the rest of the flags used here, but their meanings as well as other options can be found here

Salmon outputs

Let's take a look at what salmon spits out. The first file we will look at is a log that is found at **/logs/salmon_quant.log**. This file contains info about the quantification, but there's one line of this file in particular that we are interested in. It lets us know how many of the reads in the fastq file that salmon found a home for in the transcriptome fasta.

```
[2020-06-11 10:28:42.441] [jointLog] [info] Index contained 133618 targets
[2020-06-11 10:28:44.299] [jointLog] [info] Automatically detected most likely library type as IU
[2020-06-11 10:30:05.185] [jointLog] [info] Thread saw mini-batch with a maximum of 0.88% zero probability fragments
[2020-06-11 10:30:05.185] [jointLog] [info] Thread saw mini-batch with a maximum of 0.74% zero probability fragments
[2020-06-11 10:30:05.185] [jointLog] [info] Thread saw mini-batch with a maximum of 0.80% zero probability fragments
[2020-06-11 10:30:05.194] [jointLog] [info] Thread saw mini-batch with a maximum of 0.72% zero probability fragments
[2020-06-11 10:30:05.203] [jointLog] [info] Thread saw mini-batch with a maximum of 0.74% zero probability fragments
[2020-06-11 10:30:05.240] [jointLog] [info] Thread saw mini-batch with a maximum of 0.84% zero probability fragments
[2020-06-11 10:30:05.240] [jointLog] [info] Thread saw mini-batch with a maximum of 0.76% zero probability fragments
[2020-06-11 10:30:05.251] [jointLog] [info] Thread saw mini-batch with a maximum of 0.80% zero probability fragments
[2020-06-11 10:30:05.567] [fileLog] [info]
At end of round 0
==================
Observed 32228608 total fragments (32228608 in most recent round)

[2020-06-11 10:30:05.566] [jointLog] [info] Computed 406673 rich equivalence classes for further processing
[2020-06-11 10:30:05.566] [jointLog] [info] Counted 30081492 total reads in the equivalence classes
[2020-06-11 10:30:05.575] [jointLog] [info] Mapping rate = 93.3379%

[2020-06-11 10:30:05.575] [jointLog] [info] finished quantifyLibrary()
[2020-06-11 10:30:05.577] [jointLog] [info] Starting optimizer
[2020-06-11 10:30:05.695] [jointLog] [info] Marked 0 weighted equivalence classes as degenerate
[2020-06-11 10:30:05.714] [jointLog] [info] iteration = 0 | max rel diff. = 7200.43
[2020-06-11 10:30:05.907] [jointLog] [info] iteration 11, adjusting effective lengths to account for biases
[2020-06-11 10:30:32.535] [jointLog] [info] Computed expected counts (for bias correction)
```

There are a lot of lines in this file, but really only one that we are interested in. We want the one that tells us the "Mapping rate." How could we easily and efficiently look at the mapping rates of all our samples? **Grep!**

```
#Get the mapping rates for all samples
#In each log file, the line that we are interested in contains the string 'Mapping ' (notice the space)
grep 'Mapping ' ../data/salmonouts/*/logs/salmon_quant.log
```

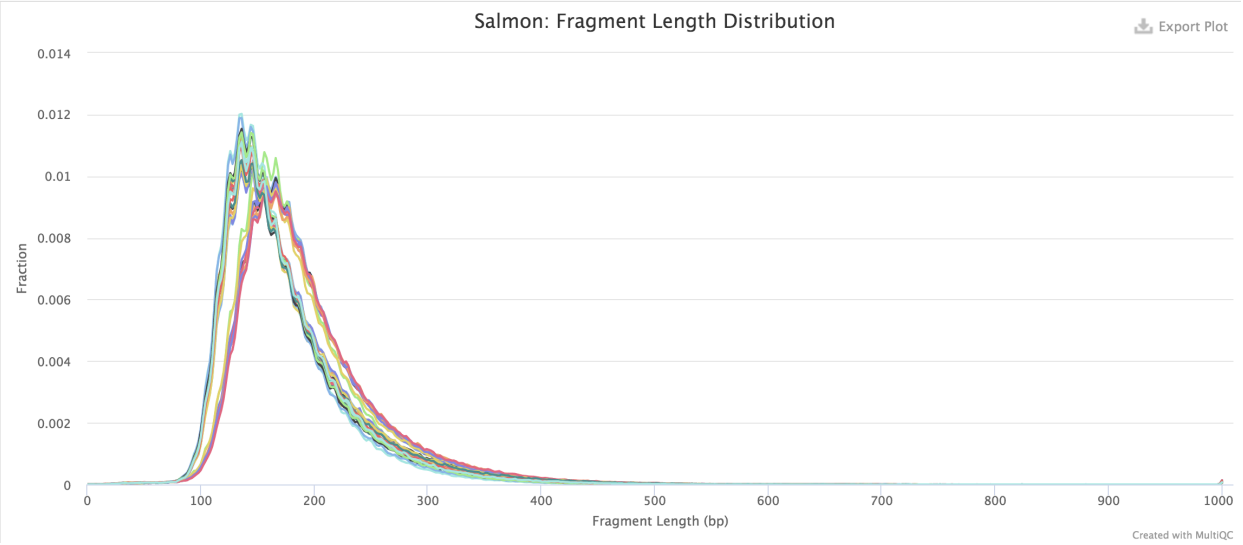
Another way to do this is to use a tool called MultiQC. MultiQC is a python package that, given a place to look, will scan for the log files produced by many common sequence analysis programs, including salmon. It will output an html file that is nice for inspecting sample stats and quickly identifying outlier samples.

```
multiqc data/salmonouts/*
```

General Statistics

Copy table | Configure Columns | Plot | Showing 30/30 rows and 2/2 columns.

Sample Name	% Aligned	M Aligned
DIV0.Rep1	93.3%	30.1
DIV0.Rep2	93.1%	31.3
DIV0.Rep3	93.0%	35.9
DIV1.Rep1	92.5%	38.4
DIV1.Rep2	92.7%	39.1
DIV1.Rep3	93.0%	25.8
DIV1.Rep4	93.4%	28.9
DIV16.Rep1	93.0%	28.2
DIV16.Rep2	93.1%	25.3
DIV16.Rep3	92.9%	29.4
DIV16.Rep4	93.7%	26.0
DIV21.Rep1	93.2%	25.4
DIV21.Rep2	93.9%	28.3
DIV21.Rep3	94.1%	28.8
DIV21.Rep4	93.0%	28.8



Moving from transcript quantifications to gene quantifications

As we discussed, salmon quantifies *transcripts*, not *genes*. However, genes are made up of transcripts, so we can calculate gene expression values from transcript expression values if we knew which transcripts belonged to which genes. We can get this information through `biomaRt`.

`biomaRt` has many tables that relate genes, transcripts, and other useful data include gene biotypes and gene ontology categories, even across species. Let's use it here to get a table of genes and transcripts for the mouse genome.

```
#Load biomaRt
library(biomaRt)

#First we need to define a 'mart' to use.  There are a handful of them that you can see here:
listMarts(mart = NULL, host = 'uswest.ensembl.org')
```

```
##                biomart                version
## 1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 101
## 2   ENSEMBL_MART_MOUSE      Mouse strains 101
## 3     ENSEMBL_MART_SNP  Ensembl Variation 101
## 4 ENSEMBL_MART_FUNCGEN Ensembl Regulation 101
```

I encourage you to see what is in each mart, but for now we are only going to use ENSEMBL_MART_ENSEMBL
This may take a minute or two to connect.

```
mart <- biomaRt::useMart("ENSEMBL_MART_ENSEMBL", host='uswest.ensembl.org')
```

Alright, we've chosen our mart. What datasets are available in this mart?

```
datasets <- listDatasets(mart)
kable(datasets)
```

dataset	description	version
acalliptera_gene_ensembl	Eastern happy genes (fAstCal1.2)	fAstCal1.2
acarolinensis_gene_ensembl	Anole lizard genes (AnoCar2.0)	AnoCar2.0
acchrysaetos_gene_ensembl	Golden eagle genes (bAquChr1.2)	bAquChr1.2
acitrinellus_gene_ensembl	Midas cichlid genes (Midas_v5)	Midas_v5
amelanoleuca_gene_ensembl	Panda genes (ailMel1)	ailMel1
amexicanus_gene_ensembl	Mexican tetra genes (Astyanax_mexicanus-2.0)	Astyanax_mexicanus-2.0
ampachon_gene_ensembl	Pachon cavefish genes (Astyanax_mexicanus-1.0.2)	Astyanax_mexicanus-1.0.2
anancymaae_gene_ensembl	Ma's night monkey genes (Anan_2.0)	Anan_2.0
aplatyrhynchos_gene_ensembl	Mallard genes (ASM874695v1)	ASM874695v1
applatyrhynchos_gene_ensembl	Duck genes (CAU_duck1.0)	CAU_duck1.0
atestudineus_gene_ensembl	Climbing perch genes (fAnaTes1.1)	fAnaTes1.1
bbbison_gene_ensembl	American bison genes (Bison_UMD1.0)	Bison_UMD1.0
bgrunniens_gene_ensembl	Domestic yak genes (LU_Bosgru_v3.0)	LU_Bosgru_v3.0
bihybrid_gene_ensembl	Hybrid - Bos Indicus genes (UOA_Brahman_1)	UOA_Brahman_1
bmutus_gene_ensembl	Wild yak genes (BosGru_v2.0)	BosGru_v2.0
bsplendens_gene_ensembl	Siamese fighting fish genes (fBetSpl5.2)	fBetSpl5.2
btaurus_gene_ensembl	Cow genes (ARS-UCD1.2)	ARS-UCD1.2
bthybrid_gene_ensembl	Hybrid - Bos Taurus genes (UOA_Angus_1)	UOA_Angus_1
cabingdonii_gene_ensembl	Abingdon island giant tortoise genes (ASM359739v1)	ASM359739v1
capalliatus_gene_ensembl	Angola colobus genes (Cang.pa_1.0)	Cang.pa_1.0
caperea_gene_ensembl	Brazilian guinea pig genes (CavAp1.0)	CavAp1.0
catys_gene_ensembl	Sooty mangabey genes (Caty_1.0)	Caty_1.0
ccanadensis_gene_ensembl	American beaver genes (C.can_genome_v1.0)	C.can_genome_v1.0
ccapucinus_gene_ensembl	Capuchin genes (Cebus_imitator-1.0)	Cebus_imitator-1.0
ccarpio_gene_ensembl	Common carp genes (common_carp_genome)	common_carp_genome
cdromedarius_gene_ensembl	Arabian camel genes (CamDro2)	CamDro2
celegans_gene_ensembl	Caenorhabditis elegans genes (WBcel235)	WBcel235
cgchok1gshd_gene_ensembl	Chinese hamster CHOK1GS genes (CHOK1GS_HDv1)	CHOK1GS_HDv1
cgcrigri_gene_ensembl	Chinese hamster CriGri genes (CriGri_1.0)	CriGri_1.0

dataset	description	version
cgobio_gene_ensembl	Channel bull blenny genes (fCotGob3.1)	fCotGob3.1
cgpicr_gene_ensembl	Chinese hamster PICR genes (CriGri-PICR)	CriGri-PICR
charengus_gene_ensembl	Atlantic herring genes (Ch_v2.0.2)	Ch_v2.0.2
chircus_gene_ensembl	Goat genes (ARS1)	ARS1
choffmanni_gene_ensembl	Sloth genes (choHof1)	choHof1
cintestinalis_gene_ensembl	C.intestinalis genes (KH)	KH
cjacchus_gene_ensembl	Marmoset genes (ASM275486v1)	ASM275486v1
cjaponica_gene_ensembl	Japanese quail genes (Coturnix_japonica_2.0)	Coturnix_japonica_2.0
clanigera_gene_ensembl	Long-tailed chinchilla genes (ChiLan1.0)	ChiLan1.0
cldingo_gene_ensembl	Dingo genes (ASM325472v1)	ASM325472v1
clfamiliaris_gene_ensembl	Dog genes (CanFam3.1)	CanFam3.1
cpbellii_gene_ensembl	Painted turtle genes (Chrysemys_picta_bellii-3.0.3)	Chrysemys_picta_bellii-3.0.3
cporcellus_gene_ensembl	Guinea Pig genes (Cavpor3.0)	Cavpor3.0
cporosus_gene_ensembl	Australian saltwater crocodile genes (CroPor_comp1)	CroPor_comp1
csabaeus_gene_ensembl	Vervet-AGM genes (ChlSab1.1)	ChlSab1.1
csavignyi_gene_ensembl	C.savignyi genes (CSAV 2.0)	CSAV 2.0
csemilaevis_gene_ensembl	Tongue sole genes (Cse_v1.0)	Cse_v1.0
csyrichta_gene_ensembl	Tarsier genes (Tarsius_syrichta-2.0.1)	Tarsius_syrichta-2.0.1
cvariegatus_gene_ensembl	Sheepshead minnow genes (C_variegatus-1.0)	C_variegatus-1.0
dclupeoides_gene_ensembl	Denticle herring genes (fDenClu1.1)	fDenClu1.1
dlabrax_gene_ensembl	European seabass genes (seabass_V1.0)	seabass_V1.0
dmelanogaster_gene_ensembl	Drosophila melanogaster genes (BDGP6.28)	BDGP6.28
dnovaehollandiae_gene_ensembl	Emu genes (droNov1)	droNov1
dnovemcinctus_gene_ensembl	Armadillo genes (Dasnov3.0)	Dasnov3.0
dordii_gene_ensembl	Kangaroo rat genes (Dord_2.0)	Dord_2.0
drerio_gene_ensembl	Zebrafish genes (GRCz11)	GRCz11
eaasinus_gene_ensembl	Donkey genes (ASM303372v1)	ASM303372v1
eburgeri_gene_ensembl	Hagfish genes (Eburgeri_3.2)	Eburgeri_3.2
ecaballus_gene_ensembl	Horse genes (EquCab3.0)	EquCab3.0
ecalabaricus_gene_ensembl	Reedfish genes (fErpCal1.1)	fErpCal1.1
eelectricus_gene_ensembl	Electric eel genes (Ee_SOAP_WITH_SSPACE)	Ee_SOAP_WITH_SSPACE
eeuropaeus_gene_ensembl	Hedgehog genes (eriEur1)	eriEur1
elucius_gene_ensembl	Northern pike genes (Eluc_v4)	Eluc_v4
enaucrates_gene_ensembl	Live sharksucker genes (fEcheNa1.1)	fEcheNa1.1
etelfairi_gene_ensembl	Lesser hedgehog tenrec genes (TENREC)	TENREC
falbicollis_gene_ensembl	Flycatcher genes (FicAlb_1.4)	FicAlb_1.4
fcatus_gene_ensembl	Cat genes (Felis_catus_9.0)	Felis_catus_9.0
fdamarensis_gene_ensembl	Damara mole rat genes (DMR_v1.0)	DMR_v1.0
fheteroclitus_gene_ensembl	Mummichog genes (Fundulus_heteroclitus-3.0.2)	Fundulus_heteroclitus-3.0.2
gaculeatus_gene_ensembl	Stickleback genes (BROAD S1)	BROAD S1
gagassizii_gene_ensembl	Agassiz's desert tortoise genes (ASM289641v1)	ASM289641v1
ggallus_gene_ensembl	Chicken genes (GRCg6a)	GRCg6a
ggorilla_gene_ensembl	Gorilla genes (gorGor4)	gorGor4
gmorhua_gene_ensembl	Cod genes (gadMor1)	gadMor1
gwilldenowi_gene_ensembl	Blunt-snouted clingfish genes (fGouWil2.1)	fGouWil2.1
hburtoni_gene_ensembl	Burton's mouthbrooder genes (AstBur1.0)	AstBur1.0
hcomes_gene_ensembl	Tiger tail seahorse genes (H_comes_QL1_v1)	H_comes_QL1_v1

dataset	description	version
hgfemale_gene_ensembl	Naked mole-rat female genes (HetGla_female_1.0)	HetGla_female_1.0
hgmale_gene_ensembl	Naked mole-rat male genes (HetGla_1.0)	HetGla_1.0
hhucho_gene_ensembl	Huchen genes (ASM331708v1)	ASM331708v1
hsapiens_gene_ensembl	Human genes (GRCh38.p13)	GRCh38.p13
ipunctatus_gene_ensembl	Channel catfish genes (IpCoco_1.2)	IpCoco_1.2
itridecemlineatus_gene_ensembl	Squirrel genes (SpeTri2.0)	SpeTri2.0
jhyemalis_gene_ensembl	Dark-eyed junco genes (ASM382977v1)	ASM382977v1
jjaculus_gene_ensembl	Lesser Egyptian jerboa genes (JacJac1.0)	JacJac1.0
lafricana_gene_ensembl	Elephant genes (Loxafr3.0)	Loxafr3.0
lbergylta_gene_ensembl	Ballan wrasse genes (BallGen_V1)	BallGen_V1
lcalcarifer_gene_ensembl	Barramundi perch genes (ASB_HGAPassembly_v1)	ASB_HGAPassembly_v1
lcanadensis_gene_ensembl	Canada lynx genes (mLynCan4_v1.p)	mLynCan4_v1.p
lchalumnae_gene_ensembl	Coelacanth genes (LatCha1)	LatCha1
lcrocea_gene_ensembl	Large yellow croaker genes (L_crocea_2.0)	L_crocea_2.0
loculatus_gene_ensembl	Spotted gar genes (LepOcu1)	LepOcu1
lsdomestica_gene_ensembl	Bengalese finch genes (LonStrDom1)	LonStrDom1
malbus_gene_ensembl	Swamp eel genes (M_albus_1.0)	M_albus_1.0
marmatus_gene_ensembl	Zig-zag eel genes (fMasArm1.1)	fMasArm1.1
mauratus_gene_ensembl	Golden Hamster genes (MesAur1.0)	MesAur1.0
mcaroli_gene_ensembl	Ryukyu mouse genes (CAROLI_EIJ_v1.1)	CAROLI_EIJ_v1.1
mdomestica_gene_ensembl	Opossum genes (ASM229v1)	ASM229v1
mfascicularis_gene_ensembl	Crab-eating macaque genes (Macaca_fascicularis_5.0)	Macaca_fascicularis_5.0
mgallopavo_gene_ensembl	Turkey genes (Turkey_2.01)	Turkey_2.01
mleucophaeus_gene_ensembl	Drill genes (Mleu.le_1.0)	Mleu.le_1.0
mlucifugus_gene_ensembl	Microbat genes (Myoluc2.0)	Myoluc2.0
mmmarmota_gene_ensembl	Alpine marmot genes (marMar2.1)	marMar2.1
mmulatta_gene_ensembl	Macaque genes (Mmul_10)	Mmul_10
mmurdjan_gene_ensembl	Pinecone soldierfish genes (fMyrMur1.1)	fMyrMur1.1
mmurinus_gene_ensembl	Mouse Lemur genes (Mmur_3.0)	Mmur_3.0
mmusculus_gene_ensembl	Mouse genes (GRCm38.p6)	GRCm38.p6
mnemestrina_gene_ensembl	Big-tailed macaque genes (Mnem_1.0)	Mnem_1.0
mochrogaster_gene_ensembl	Prairie vole genes (MicOch1.0)	MicOch1.0
mpahari_gene_ensembl	Shrew mouse genes (PAHARI_EIJ_v1.1)	PAHARI_EIJ_v1.1
mpfuro_gene_ensembl	Ferret genes (MusPutFur1.0)	MusPutFur1.0
mspicilegus_gene_ensembl	Steppe mouse genes (MUSP714)	MUSP714
mspretus_gene_ensembl	Algerian mouse genes (SPRET_EiJ_v1)	SPRET_EiJ_v1
mundulatus_gene_ensembl	Budgerigar genes (Melopsittacus_undulatus_6.3)	Melopsittacus_undulatus_6.3
munguiculatus_gene_ensembl	Mongolian gerbil genes (MunDraft-v1.0)	MunDraft-v1.0
mvitellinus_gene_ensembl	Golden-collared manakin genes (ASM171598v2)	ASM171598v2
mzebra_gene_ensembl	Zebra mbuna genes (M_zebra_UMD2a)	M_zebra_UMD2a
nbrichardi_gene_ensembl	Lyretail cichlid genes (NeoBri1.0)	NeoBri1.0
neugenii_gene_ensembl	Wallaby genes (Meug_1.0)	Meug_1.0
ngalili_gene_ensembl	Upper Galilee mountains blind mole rat genes (S.galili_v1.0)	S.galili_v1.0
nleucogenys_gene_ensembl	Gibbon genes (Nleu_3.0)	Nleu_3.0
nvison_gene_ensembl	American mink genes (NNQGG.v01)	NNQGG.v01
oanatinus_gene_ensembl	Platypus genes (mOrnAna1.p.v1)	mOrnAna1.p.v1
oaries_gene_ensembl	Sheep (texel) genes (Oar_v3.1)	Oar_v3.1

dataset	description	version
oaureus_gene_ensembl	Blue tilapia genes (ASM587006v1)	ASM587006v1
ocuniculus_gene_ensembl	Rabbit genes (OryCun2.0)	OryCun2.0
odegus_gene_ensembl	Degu genes (OctDeg1.0)	OctDeg1.0
ogarnettii_gene_ensembl	Bushbaby genes (OtoGar3)	OtoGar3
olatipes_gene_ensembl	Japanese medaka HdrR genes (ASM223467v1)	ASM223467v1
olhni_gene_ensembl	Japanese medaka HNI genes (ASM223471v1)	ASM223471v1
olhsok_gene_ensembl	Japanese medaka HSOK genes (ASM223469v1)	ASM223469v1
omelastigma_gene_ensembl	Indian medaka genes (Om_v0.7.RACA)	Om_v0.7.RACA
omykiss_gene_ensembl	Rainbow trout genes (Omyk_1.0)	Omyk_1.0
oniloticus_gene_ensembl	Nile tilapia genes (O_niloticus_UMD_NMBU)	O_niloticus_UMD_NMBU
oprinceps_gene_ensembl	Pika genes (OchPri2.0-Ens)	OchPri2.0-Ens
pabelii_gene_ensembl	Orangutan genes (PPYG2)	PPYG2
panubis_gene_ensembl	Olive baboon genes (Panu_3.0)	Panu_3.0
pcapensis_gene_ensembl	Hyrax genes (proCap1)	proCap1
pcinereus_gene_ensembl	Koala genes (phaCin_unsw_v4.1)	phaCin_unsw_v4.1
pcoquereli_gene_ensembl	Coquerel's sifaka genes (Pcoq_1.0)	Pcoq_1.0
pformosa_gene_ensembl	Amazon molly genes (Poecilia_formosa-5.1.2)	Poecilia_formosa-5.1.2
platipinna_gene_ensembl	Sailfin molly genes (P_latipinna-1.0)	P_latipinna-1.0
pmarinus_gene_ensembl	Lamprey genes (Pmarinus_7.0)	Pmarinus_7.0
pmbairdii_gene_ensembl	Northern American deer mouse genes (HU_Pman_2.1)	HU_Pman_2.1
pmexicana_gene_ensembl	Shortfin molly genes (P_mexicana-1.0)	P_mexicana-1.0
pnattereri_gene_ensembl	Red-bellied piranha genes (Pygocentrus_nattereri-1.0.2)	Pygocentrus_nattereri-1.0.2
pnyererei_gene_ensembl	Makobe Island cichlid genes (PunNye1.0)	PunNye1.0
ppaniscus_gene_ensembl	Bonobo genes (panpan1.1)	panpan1.1
ppardus_gene_ensembl	Leopard genes (PanPar1.0)	PanPar1.0
pranga_gene_ensembl	Indian glassy fish genes (fParRan2.1)	fParRan2.1
preticulata_gene_ensembl	Guppy genes (Guppy_female_1.0_MT)	Guppy_female_1.0_MT
psimus_gene_ensembl	Greater bamboo lemur genes (Prosim_1.0)	Prosim_1.0
psinensis_gene_ensembl	Chinese softshell turtle genes (PelSin_1.0)	PelSin_1.0
ptaltaica_gene_ensembl	Tiger genes (PanTig1.0)	PanTig1.0
ptephrosceles_gene_ensembl	Ugandan red Colobus genes (ASM277652v2)	ASM277652v2
ptroglodytes_gene_ensembl	Chimpanzee genes (Pan_tro_3.0)	Pan_tro_3.0
pvampyrus_gene_ensembl	Megabat genes (pteVam1)	pteVam1
pvitticeps_gene_ensembl	Central bearded dragon genes (pvi1.1)	pvi1.1
rbieti_gene_ensembl	Black snub-nosed monkey genes (ASM169854v1)	ASM169854v1
rferrumequinum_gene_ensembl	Greater horseshoe bat genes (mRhiFer1_v1.p)	mRhiFer1_v1.p
rnorvegicus_gene_ensembl	Rat genes (Rnor_6.0)	Rnor_6.0
rroxellana_gene_ensembl	Golden snub-nosed monkey genes (Rrox_v1)	Rrox_v1
saraneus_gene_ensembl	Shrew genes (sorAra1)	sorAra1
saurata_gene_ensembl	Gilthead seabream genes (fSpaAur1.1)	fSpaAur1.1
sboliviensis_gene_ensembl	Bolivian squirrel monkey genes (SaiBol1.0)	SaiBol1.0
scanaria_gene_ensembl	Common canary genes (SCA1)	SCA1
scerevisiae_gene_ensembl	Saccharomyces cerevisiae genes (R64-1-1)	R64-1-1
sdumerili_gene_ensembl	Greater amberjack genes (Sdu_1.0)	Sdu_1.0
sfasciatus_gene_ensembl	Jewelled blenny genes (fSalaFa1.1)	fSalaFa1.1
sformosus_gene_ensembl	Asian bonytongue genes (fSclFor1.1)	fSclFor1.1
shabroptila_gene_ensembl	Kakapo genes (bStrHab1_v1.p)	bStrHab1_v1.p
sharrisii_gene_ensembl	Tasmanian devil genes (Devil_ref v7.0)	Devil_ref v7.0
sldorsalis_gene_ensembl	Yellowtail amberjack genes (Sedor1)	Sedor1

dataset	description	version
smaximus_gene_ensembl	Turbot genes (ASM318616v1)	ASM318616v1
smerianae_gene_ensembl	Argentine black and white tegu genes (HLtupMer3)	HLtupMer3
sorbicularis_gene_ensembl	Orbiculate cardinalfish genes (fSphaOr1.1)	fSphaOr1.1
spunctatus_gene_ensembl	Tuatara genes (ASM311381v1)	ASM311381v1
ssalar_gene_ensembl	Atlantic salmon genes (ICSASG_v2)	ICSASG_v2
ssbamei_gene_ensembl	Pig - Bamei genes (Bamei_pig_v1)	Bamei_pig_v1
ssberkshire_gene_ensembl	Pig - Berkshire genes (Berkshire_pig_v1)	Berkshire_pig_v1
sscrofa_gene_ensembl	Pig genes (Sscrofa11.1)	Sscrofa11.1
sshampshire_gene_ensembl	Pig - Hampshire genes (Hampshire_pig_v1)	Hampshire_pig_v1
ssjinhua_gene_ensembl	Pig - Jinhua genes (Jinhua_pig_v1)	Jinhua_pig_v1
sslandrace_gene_ensembl	Pig - Landrace genes (Landrace_pig_v1)	Landrace_pig_v1
sslargewhite_gene_ensembl	Pig - Largewhite genes (Large_White_v1)	Large_White_v1
ssmeishan_gene_ensembl	Pig - Meishan genes (Meishan_pig_v1)	Meishan_pig_v1
sspietrain_gene_ensembl	Pig - Pietrain genes (Pietrain_pig_v1)	Pietrain_pig_v1
ssrongchang_gene_ensembl	Pig - Rongchang genes (Rongchang_pig_v1)	Rongchang_pig_v1
sstibetan_gene_ensembl	Pig - Tibetan genes (Tibetan_Pig_v2)	Tibetan_Pig_v2
ssusmarc_gene_ensembl	Pig USMARC genes (USMARCv1.0)	USMARCv1.0
sswuzhishan_gene_ensembl	Pig - Wuzhishan genes (minipig_v1.0)	minipig_v1.0
strutta_gene_ensembl	Brown trout genes (fSalTru1.1)	fSalTru1.1
tbelangeri_gene_ensembl	Tree Shrew genes (tupBel1)	tupBel1
tgelada_gene_ensembl	Gelada genes (Tgel_1.0)	Tgel_1.0
tguttata_gene_ensembl	Zebra finch genes (bTaeGut1_v1.p)	bTaeGut1_v1.p
tnigroviridis_gene_ensembl	Tetraodon genes (TETRAODON 8.0)	TETRAODON 8.0
trubripes_gene_ensembl	Fugu genes (fTakRub1.2)	fTakRub1.2
ttruncatus_gene_ensembl	Dolphin genes (turTru1)	turTru1
uamericanus_gene_ensembl	American black bear genes (ASM334442v1)	ASM334442v1
umaritimus_gene_ensembl	Polar bear genes (UrsMar_1.0)	UrsMar_1.0
vpacos_gene_ensembl	Alpaca genes (vicPac1)	vicPac1
vursinus_gene_ensembl	Common wombat genes (bare-nosed_wombat_genome_assembly)	bare-nosed_wombat_genome_assembly
vvulpes_gene_ensembl	Red fox genes (VulVul2.2)	VulVul2.2
xtropicalis_gene_ensembl	Tropical clawed frog genes (Xenopus_tropicalis_v9.1)	Xenopus_tropicalis_v9.1

Alot of stuff for a lot of species! Perhaps we want to limit it to see which ones are relevant to mouse.

```
mousedatasets <- filter(datasets, grepl('mmusculus', dataset))
head(mousedatasets)
```

```
##                  dataset             description   version
## 1 mmusculus_gene_ensembl Mouse genes (GRCm38.p6) GRCm38.p6
```

Ah so we probably want the dataset called 'mmusculus_gene_ensembl'!

```
mart <- biomaRt::useMart("ENSEMBL_MART_ENSEMBL", dataset = "mmusculus_gene_ensembl", host='uswest.ensembl
```

OK what goodies are in this dataset?

```
goodies <- listAttributes(mart)
head(goodies)
```

```
##                          name                 description         page
## 1              ensembl_gene_id              Gene stable ID feature_page
```

```
## 2       ensembl_gene_id_version       Gene stable ID version feature_page
## 3         ensembl_transcript_id         Transcript stable ID feature_page
## 4 ensembl_transcript_id_version Transcript stable ID version feature_page
## 5            ensembl_peptide_id            Protein stable ID feature_page
## 6    ensembl_peptide_id_version    Protein stable ID version feature_page
```

So there are 2885 rows of goodies about the mouse genome and its relationship to *many* other genomes. However, you can probably see that the ones that are most useful to us right now are right at the top: 'ensembl_transcript_id' and 'ensembl_gene_id'. We can use those attributes in our mart to make a table relating genes and transcripts.

I'm going to through one more attribute in: external_gene_name. Those are usually more informative than ensembl IDs.

```
t2g <- biomaRt::getBM(attributes = c('ensembl_transcript_id', 'ensembl_gene_id', 'external_gene_name'),
```

```
## Cache found
```

```
head(t2g)
```

```
##   ensembl_transcript_id    ensembl_gene_id external_gene_name
## 1    ENSMUST00000082423 ENSMUSG00000064372              mt-Tp
## 2    ENSMUST00000082422 ENSMUSG00000064371              mt-Tt
## 3    ENSMUST00000082421 ENSMUSG00000064370            mt-Cytb
## 4    ENSMUST00000082420 ENSMUSG00000064369              mt-Te
## 5    ENSMUST00000082419 ENSMUSG00000064368             mt-Nd6
## 6    ENSMUST00000082418 ENSMUSG00000064367             mt-Nd5
```

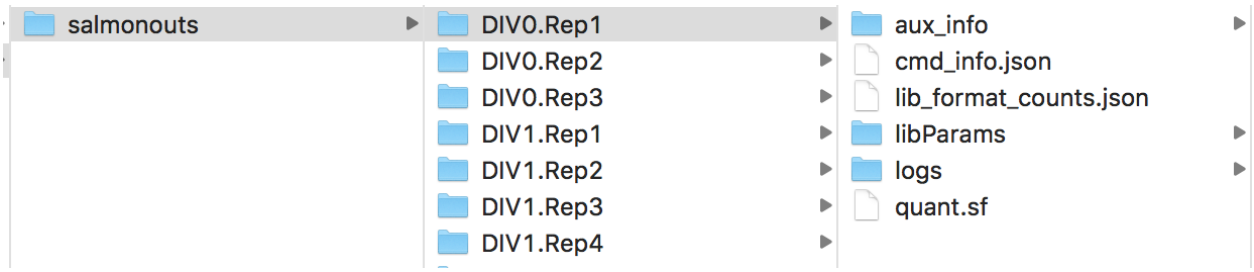
Alright this looks good! We are going to split this into two tables. One that contains transcript ID and gene ID, and the other that contains gene ID and gene name.

```
geneid2name <- dplyr::select(t2g, c(ensembl_gene_id, ensembl_transcript_id))
t2g <- dplyr::select(t2g, c(ensembl_transcript_id, ensembl_gene_id))
```

Getting gene level expression data with `tximport`

Now that we have our table relating transcripts and genes, we can give it to tximport to have it calculate gene-level expression data from our transcript-level expression data.

First, we have to tell it where the salmon quantification files (the quant.sf files) are. Here's what our directory structure that contains these files looks like:




```
#The directory where all of the sample-specific salmon subdirectories live
base_dir <- '../data/salmonouts/'

#The names of all the sample-specific salmon subdirectories
sample_ids <- c('DIVminus8.Rep1', 'DIVminus8.Rep2', 'DIVminus8.Rep3', 'DIVminus8.Rep4',
                'DIVminus4.Rep1', 'DIVminus4.Rep2', 'DIVminus4.Rep3',
                'DIV0.Rep1', 'DIV0.Rep2', 'DIV0.Rep3',
```

```
                'DIV1.Rep1', 'DIV1.Rep2', 'DIV1.Rep3', 'DIV1.Rep4',
                'DIV7.Rep1', 'DIV7.Rep2', 'DIV7.Rep3', 'DIV7.Rep4',
                'DIV16.Rep1', 'DIV16.Rep2', 'DIV16.Rep3', 'DIV16.Rep4',
                'DIV21.Rep1', 'DIV21.Rep2', 'DIV21.Rep3', 'DIV21.Rep4',
                'DIV28.Rep1', 'DIV28.Rep2', 'DIV28.Rep3', 'DIV28.Rep4')

#So what we want to do now is create paths to each quant.sf file that is in each sample_id.
#This can be done by combining the base_dir, each sample_id directory, and 'quant.sf'
#For example, the path to the first file will be data/salmonouts/DIVminus8.Rep1/quant.sf

salm_dirs <- sapply(sample_ids, function(id) file.path(base_dir, id, 'quant.sf'))
salm_dirs
```

```
##                                DIVminus8.Rep1
## "../data/salmonouts//DIVminus8.Rep1/quant.sf"
##                                DIVminus8.Rep2
## "../data/salmonouts//DIVminus8.Rep2/quant.sf"
##                                DIVminus8.Rep3
## "../data/salmonouts//DIVminus8.Rep3/quant.sf"
##                                DIVminus8.Rep4
## "../data/salmonouts//DIVminus8.Rep4/quant.sf"
##                                DIVminus4.Rep1
## "../data/salmonouts//DIVminus4.Rep1/quant.sf"
##                                DIVminus4.Rep2
## "../data/salmonouts//DIVminus4.Rep2/quant.sf"
##                                DIVminus4.Rep3
## "../data/salmonouts//DIVminus4.Rep3/quant.sf"
##                                     DIV0.Rep1
##      "../data/salmonouts//DIV0.Rep1/quant.sf"
##                                     DIV0.Rep2
##      "../data/salmonouts//DIV0.Rep2/quant.sf"
##                                     DIV0.Rep3
##      "../data/salmonouts//DIV0.Rep3/quant.sf"
##                                     DIV1.Rep1
##      "../data/salmonouts//DIV1.Rep1/quant.sf"
##                                     DIV1.Rep2
##      "../data/salmonouts//DIV1.Rep2/quant.sf"
##                                     DIV1.Rep3
##      "../data/salmonouts//DIV1.Rep3/quant.sf"
##                                     DIV1.Rep4
##      "../data/salmonouts//DIV1.Rep4/quant.sf"
##                                     DIV7.Rep1
##      "../data/salmonouts//DIV7.Rep1/quant.sf"
##                                     DIV7.Rep2
##      "../data/salmonouts//DIV7.Rep2/quant.sf"
##                                     DIV7.Rep3
##      "../data/salmonouts//DIV7.Rep3/quant.sf"
##                                     DIV7.Rep4
##      "../data/salmonouts//DIV7.Rep4/quant.sf"
##                                    DIV16.Rep1
##     "../data/salmonouts//DIV16.Rep1/quant.sf"
##                                    DIV16.Rep2
##     "../data/salmonouts//DIV16.Rep2/quant.sf"
##                                    DIV16.Rep3
```

```
##     "../data/salmonouts//DIV16.Rep3/quant.sf"
##                                     DIV16.Rep4
##     "../data/salmonouts//DIV16.Rep4/quant.sf"
##                                     DIV21.Rep1
##     "../data/salmonouts//DIV21.Rep1/quant.sf"
##                                     DIV21.Rep2
##     "../data/salmonouts//DIV21.Rep2/quant.sf"
##                                     DIV21.Rep3
##     "../data/salmonouts//DIV21.Rep3/quant.sf"
##                                     DIV21.Rep4
##     "../data/salmonouts//DIV21.Rep4/quant.sf"
##                                     DIV28.Rep1
##     "../data/salmonouts//DIV28.Rep1/quant.sf"
##                                     DIV28.Rep2
##     "../data/salmonouts//DIV28.Rep2/quant.sf"
##                                     DIV28.Rep3
##     "../data/salmonouts//DIV28.Rep3/quant.sf"
##                                     DIV28.Rep4
##     "../data/salmonouts//DIV28.Rep4/quant.sf"
```

You can see that we get a list of sample names and the absolute path to each sample's quantification file.

Now we are ready to run `tximport`! `tximport` is going to want paths to all the quantification files (salm_dirs) and a table that relates transcripts to genes (t2g). Luckily, we happen to have those exact two things.

```
txi <- tximport(salm_dirs, type = 'salmon', tx2gene = t2g, dropInfReps = TRUE, countsFromAbundance = 'l
```

```
## reading in files with read_tsv
```

```
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
## transcripts missing from tx2gene: 602
## summarizing abundance
## summarizing counts
## summarizing length
```

Notice how we chose *lengthscaledTPM* for our abundance measurement. This is going to give us TPM values (transcripts per million) for expression in the $abundance slot. Let's check out what we have now.

```
tpms <- txi$abundance %>%
  as.data.frame(.) %>%
  rownames_to_column(var = 'ensembl_gene_id')

head(tpms)
```

```
##      ensembl_gene_id DIVminus8.Rep1 DIVminus8.Rep2 DIVminus8.Rep3
## 1 ENSMUSG00000000001      92.799612      93.563524      95.500337
## 2 ENSMUSG00000000003       0.000000       0.000000       0.000000
## 3 ENSMUSG00000000028      59.718765      57.981259      57.935839
## 4 ENSMUSG00000000031       0.177770       0.215726       0.320948
## 5 ENSMUSG00000000037       3.187886       2.507155       2.624140
## 6 ENSMUSG00000000049       0.000000       0.000000       0.000000
##   DIVminus8.Rep4 DIVminus4.Rep1 DIVminus4.Rep2 DIVminus4.Rep3  DIV0.Rep1
## 1     101.951963     106.634101     106.395094     109.997598 148.004511
## 2       0.000000       0.000000       0.000000       0.000000   0.000000
## 3      41.663343      45.614077      47.851930      47.516125  34.180035
## 4      36.535428       0.914481       1.402704       1.317939   7.213808
## 5       2.281871       2.196733       2.309187       2.213591   5.877237
```

```
## 6       0.000000       0.000000       0.000000       0.000000   0.000000
##    DIV0.Rep2  DIV0.Rep3  DIV1.Rep1  DIV1.Rep2  DIV1.Rep3  DIV1.Rep4  DIV7.Rep1
## 1 145.539529 164.680183 142.713005 138.042672 137.027499 123.894516  76.088940
## 2   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
## 3  36.215924  39.486489  16.894844  15.391383  18.171282  16.654062   6.582942
## 4   6.431925  13.253322   3.301713   2.539161   3.591509   2.798164 151.244212
## 5   5.001733   7.579149   2.861842   3.176129   2.457556   3.591170   2.750903
## 6   0.046208   0.000000   0.000000   0.000000   0.000000   0.411807   0.188254
##    DIV7.Rep2  DIV7.Rep3  DIV7.Rep4 DIV16.Rep1 DIV16.Rep2 DIV16.Rep3 DIV16.Rep4
## 1  76.669016  79.505738  76.038479  41.689798  41.554126  41.002841  39.522314
## 2   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
## 3   6.752942   5.547107   5.510259   1.067147   1.270428   1.109391   1.136970
## 4 137.227925 169.762968 150.599359  93.312776  56.531835  63.151714  68.565195
## 5   1.827190   2.446732   2.894751   0.319281   0.708167   0.578467   0.513434
## 6   0.050654   0.277837   0.189814   0.213589   0.118890   1.024154   0.628760
##   DIV21.Rep1 DIV21.Rep2 DIV21.Rep3 DIV21.Rep4 DIV28.Rep1 DIV28.Rep2 DIV28.Rep3
## 1  28.162815  29.694158  29.333682  27.956845  17.960041  17.979483  18.102228
## 2   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
## 3   1.939558   1.003108   1.245186   1.086912   1.303824   0.935135   0.864451
## 4  32.192377  37.465959  40.725811  24.780757  67.818115  64.333586  30.408290
## 5   0.186198   0.260270   0.258075   0.585622   0.615895   0.133394   0.197050
## 6   0.303218   0.449928   0.262019   0.105047   0.567535   0.649720   0.515246
##   DIV28.Rep4
## 1  19.590551
## 2   0.000000
## 3   0.862209
## 4  60.198406
## 5   1.005331
## 6   0.360877
```

Alright, not bad!

Let's stop and think for a minute about what `tximport` did and the metric we are using (TPM). What does *transcripts per million* mean? Well, it means pretty much what it sounds like. For every million transcripts in the cell, X of them are this particular transcript. Importantly, this means when this TPM value was calculated from the number of *counts* a transcript received, this number had to be adjusted for both the total number of counts in the library and the length of a transcript.

If sample A had twice the number of total counts as sample B (i.e. was sequenced twice as deeply), then you would expect every transcript to have approximately twice the number of counts in sample A as it has in sample B. Similarly, if transcript X is twice as long as transcript Y, then you would expect that if they were equally expressed (i.e. the same number of transcript X and transcript Y molecules were present in the sample) that X would have approximately twice the counts that Y does. Working with expression units of TPM incorporates both of these normalizations.

So, if a TPM of X means that for every million transcripts in the sample that X of them were the transcript of interest, then the sum of TPM values across all species should equal one million, right?

Let's check and see if that's true.

```
sum(tpms$DIVminus8.Rep1)
```

```
## [1] 995216.3
```

```
sum(tpms$DIVminus8.Rep2)
```

```
## [1] 995244.1
```

```
sum(tpms$DIVminus8.Rep3)
```

```
## [1] 995222.4
```

OK, not quite one million, but pretty darn close.

This notion that TPMs represent proportions of a whole also leads to another interesting insight into what `tximport` is doing here. If all transcripts belong to genes, then the TPM for a gene must be the sum of the TPMs of its transcripts. Can we verify that that is true?

```
#Redefine for clarity in comparisons
tpms.genes <- tpms

#Make a new tximport object, but this time instead of giving gene expression values, give transcript ex
#This is controlled by the `txOut` argument
txi.transcripts <- tximport(salm_dirs, type = 'salmon', tx2gene = t2g, dropInfReps = TRUE,
                            countsFromAbundance = 'lengthScaledTPM', txOut = TRUE)
```

```
## reading in files with read_tsv
```

```
## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

```
#Make a table of tpm values for every transcript
tpms.txs <- txi.transcripts$abundance %>%
  as.data.frame(.) %>%
  rownames_to_column(var = 'ensembl_transcript_id') %>%
  inner_join(t2g, ., by = 'ensembl_transcript_id')

head(tpms.genes)
```

```
##      ensembl_gene_id DIVminus8.Rep1 DIVminus8.Rep2 DIVminus8.Rep3
## 1 ENSMUSG00000000001      92.799612      93.563524      95.500337
## 2 ENSMUSG00000000003       0.000000       0.000000       0.000000
## 3 ENSMUSG00000000028      59.718765      57.981259      57.935839
## 4 ENSMUSG00000000031       0.177770       0.215726       0.320948
## 5 ENSMUSG00000000037       3.187886       2.507155       2.624140
## 6 ENSMUSG00000000049       0.000000       0.000000       0.000000
##   DIVminus8.Rep4 DIVminus4.Rep1 DIVminus4.Rep2 DIVminus4.Rep3  DIV0.Rep1
## 1     101.951963     106.634101     106.395094     109.997598 148.004511
## 2       0.000000       0.000000       0.000000       0.000000   0.000000
## 3      41.663343      45.614077      47.851930      47.516125  34.180035
## 4      36.535428       0.914481       1.402704       1.317939   7.213808
## 5       2.281871       2.196733       2.309187       2.213591   5.877237
## 6       0.000000       0.000000       0.000000       0.000000   0.000000
##    DIV0.Rep2  DIV0.Rep3  DIV1.Rep1  DIV1.Rep2  DIV1.Rep3  DIV1.Rep4  DIV7.Rep1
## 1 145.539529 164.680183 142.713005 138.042672 137.027499 123.894516  76.088940
## 2   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
## 3  36.215924  39.486489  16.894844  15.391383  18.171282  16.654062   6.582942
## 4   6.431925  13.253322   3.301713   2.539161   3.591509   2.798164 151.244212
## 5   5.001733   7.579149   2.861842   3.176129   2.457556   3.591170   2.750903
## 6   0.046208   0.000000   0.000000   0.000000   0.000000   0.411807   0.188254
##    DIV7.Rep2  DIV7.Rep3  DIV7.Rep4 DIV16.Rep1 DIV16.Rep2 DIV16.Rep3 DIV16.Rep4
## 1  76.669016  79.505738  76.038479  41.689798  41.554126  41.002841  39.522314
## 2   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
## 3   6.752942   5.547107   5.510259   1.067147   1.270428   1.109391   1.136970
## 4 137.227925 169.762968 150.599359  93.312776  56.531835  63.151714  68.565195
## 5   1.827190   2.446732   2.894751   0.319281   0.708167   0.578467   0.513434
```

```
## 6   0.050654   0.277837   0.189814   0.213589   0.118890   1.024154   0.628760
##   DIV21.Rep1 DIV21.Rep2 DIV21.Rep3 DIV21.Rep4 DIV28.Rep1 DIV28.Rep2 DIV28.Rep3
## 1  28.162815  29.694158  29.333682  27.956845  17.960041  17.979483  18.102228
## 2   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
## 3   1.939558   1.003108   1.245186   1.086912   1.303824   0.935135   0.864451
## 4  32.192377  37.465959  40.725811  24.780757  67.818115  64.333586  30.408290
## 5   0.186198   0.260270   0.258075   0.585622   0.615895   0.133394   0.197050
## 6   0.303218   0.449928   0.262019   0.105047   0.567535   0.649720   0.515246
##   DIV28.Rep4
## 1  19.590551
## 2   0.000000
## 3   0.862209
## 4  60.198406
## 5   1.005331
## 6   0.360877
```

```
head(tpms.txs)
```

```
##   ensembl_transcript_id    ensembl_gene_id DIVminus8.Rep1 DIVminus8.Rep2
## 1    ENSMUST00000082423 ENSMUSG00000064372     214.839964     199.081786
## 2    ENSMUST00000082422 ENSMUSG00000064371      16.850193       6.221307
## 3    ENSMUST00000082421 ENSMUSG00000064370    1426.656406    1506.843160
## 4    ENSMUST00000082420 ENSMUSG00000064369       4.044046       7.949446
## 5    ENSMUST00000082419 ENSMUSG00000064368    1082.003939     991.205932
## 6    ENSMUST00000082418 ENSMUSG00000064367     602.098536     556.061985
##   DIVminus8.Rep3 DIVminus8.Rep4 DIVminus4.Rep1 DIVminus4.Rep2 DIVminus4.Rep3
## 1      152.87167     270.905168       85.77435     165.754533       99.895723
## 2        0.00000       3.954820       15.14097       5.262049       9.292625
## 3     1505.51196    1130.763861     3794.07442    4161.758694    3226.140151
## 4       14.04814       3.790036       50.45550      30.309400      28.992991
## 5      984.39754     864.461624     3135.55056    3332.998105    2759.071050
## 6      561.25950     391.812577     1866.89956    2027.322843    1586.511988
##    DIV0.Rep1  DIV0.Rep2  DIV0.Rep3  DIV1.Rep1  DIV1.Rep2  DIV1.Rep3  DIV1.Rep4
## 1  187.78775  198.27711  173.69683  270.88804  382.64170  253.64002  275.90413
## 2   21.83537   27.75879   15.10407   13.64184   11.53694   13.93627   16.09441
## 3 5206.54231 5008.89226 4301.34043 3934.24467 4065.47074 3956.99705 3958.03279
## 4   50.30965   39.65542   36.24977   11.22529   33.22637   29.48056   42.00464
## 5 3458.74886 3386.12662 2822.65803 2765.07547 2747.08872 2540.41471 2525.49525
## 6 1967.95677 1824.92720 1599.97253 1436.62924 1481.26167 1489.54987 1277.77587
##     DIV7.Rep1   DIV7.Rep2  DIV7.Rep3   DIV7.Rep4 DIV16.Rep1 DIV16.Rep2
## 1 1461.640972 1395.690779 1376.49355 1529.665646 4587.57897 5181.23989
## 2    4.567628    4.880038    0.00000    2.296795   15.48114   23.78079
## 3 5690.831834 5502.540956 5523.44964 5826.771376 7705.90761 6978.50807
## 4   28.452516   46.767028   42.82921   37.418623   82.56610   54.12607
## 5 3741.254543 3730.882171 3970.24693 4046.412137 6601.86484 5973.19073
## 6 2005.895514 2004.343121 2012.71565 2105.629391 3155.97954 3222.69924
##   DIV16.Rep3 DIV16.Rep4 DIV21.Rep1 DIV21.Rep2 DIV21.Rep3 DIV21.Rep4 DIV28.Rep1
## 1 5561.80037 4261.57908 3982.86786 3956.74121 4060.32734 4359.61076 3064.44760
## 2   51.27327   16.86641   11.55292   30.16575   41.48171   19.32238   27.90474
## 3 7324.81129 7775.82658 7536.10304 8014.18870 8816.97052 8937.80297 9359.87704
## 4   74.85663   80.95876   52.58984   36.19890   39.82244   74.87420   65.95665
## 5 6383.83760 6875.98970 5216.80715 5410.38560 6192.39286 5596.26967 6219.18349
## 6 3395.67895 3451.94344 2837.94673 2962.35866 3359.58354 3106.80872 3522.66441
##   DIV28.Rep2 DIV28.Rep3  DIV28.Rep4
## 1 3407.31558 3864.67441  3154.48387
```

```
## 2   50.56793   56.24266    33.43132
## 3 9490.67746 9528.62802 10465.75651
## 4   64.61458   80.34666    71.63854
## 5 6942.61410 7000.38960  7166.91243
## 6 3014.16632 3660.37326  3857.41639
```

OK so lets look at the expression of ENSMUSG00000020634 in the first sample (DIVminus8.Rep1).

```
#Get sum of tpm values for transcripts that belong to ENSMUSG00000020634
tpms.tx.ENSMUSG00000020634 <- filter(tpms.txs, ensembl_gene_id == 'ENSMUSG00000020634')
sumoftxtpm <- sum(tpms.tx.ENSMUSG00000020634$DIVminus8.Rep1)

#Get gene level tpm value of ENSMUSG00000020634
genetpm <- filter(tpms.genes, ensembl_gene_id == 'ENSMUSG00000020634')$DIVminus8.Rep1

#Are they the same?
sumoftxtpm
```

```
## [1] 46.71414
```

```
genetpm
```

```
## [1] 46.71414
```

Basic RNAseq QC

OK now that we've got expression values for all genes, we now might want to use these expression values to learn a little bit about our samples. One simple question is

`Are replicates similar to each other, or at least more similar to each other than to other samples?`

If our data is worth anything at all, we would hope that differences between replicates, which are supposed to be drawn from the same condition, are smaller than differences between samples drawn from different conditions. If that's not true, it could indicate that one replicate is very different from other replciates (in which case we might want to remove it), or that the data in general is of poor quality.

Another question is

`How similar is each sample to every other sample?`

In our timecourse, we might expect that samples drawn from adjacent timepoints might be more similar to each other than samples from more distant timepoints.

Hierarchical clustering

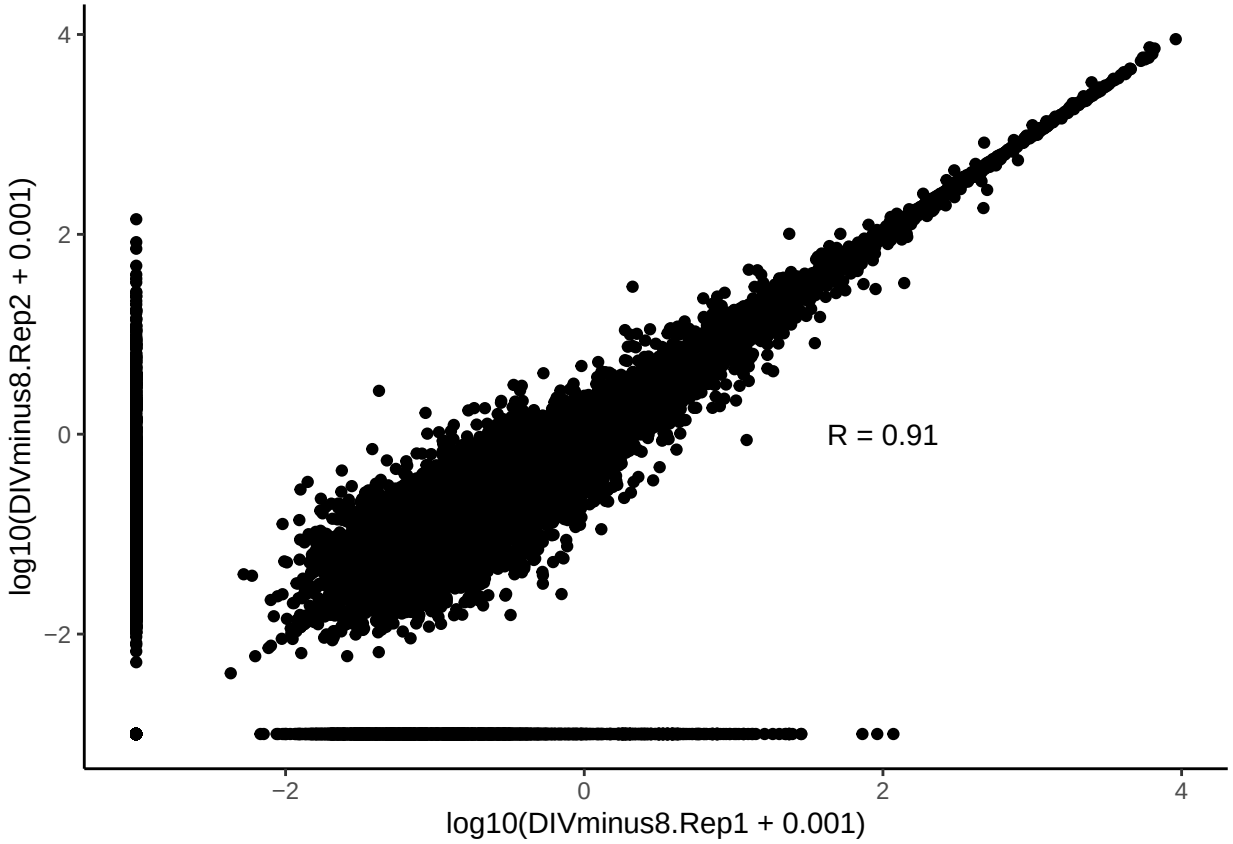
A simple way to think about this is to simply correlate TPM values for genes between samples. For plotting purposes here, let's plot the log(TPM) of two samples against each other. However, for the actual correlation coefficient we are going to be using the *Spearman* correlation method, which uses ranks, not absolute values. This means that whether or not you take the log will have no effect on the Spearman correlation coefficient.

```
#DIVminus8.Rep1 vs DIVminus8.Rep2

#Since we are plotting log TPM values, we need to add a pseudocount to all samples.
#log(0) is a problem.

#Add pseudocounts and take log within ggplot function call
r.spearman <- cor.test(tpms$DIVminus8.Rep1, tpms$DIVminus8.Rep2, method = 'spearman')$estimate[[1]]
```

```
## Warning in cor.test.default(tpms$DIVminus8.Rep1, tpms$DIVminus8.Rep2, method =
```

```
## "spearman"): Cannot compute exact p-value with ties
```

```
r.spearman <- signif(r.spearman, 2)
ggplot(tpms, aes(x = log10(DIVminus8.Rep1 + 1e-3), y = log10(DIVminus8.Rep2 + 1e-3))) + geom_point() + t
  annotate('text', x = 2, y = 0, label = paste0('R = ', r.spearman))
```



With RNAseq data, the variance of a gene's expression increases as the expression increases. However, using a pseudocount and taking the log of the expression value actually reverses this trend. Now, genes with the lowest expression have the most variance. Why is this a problem? Well, the genes with the most variance are going to be the ones that contribute the most to intersample differences. Ideally, we would like to therefore remove the relationship between expression and variance.

There are transformations, notably `rlog` and `vst`, that are made to deal with this, but they are best used when dealing with normalized **count** data, while here we are dealing with TPMs. We will talk about counts later, but not here.

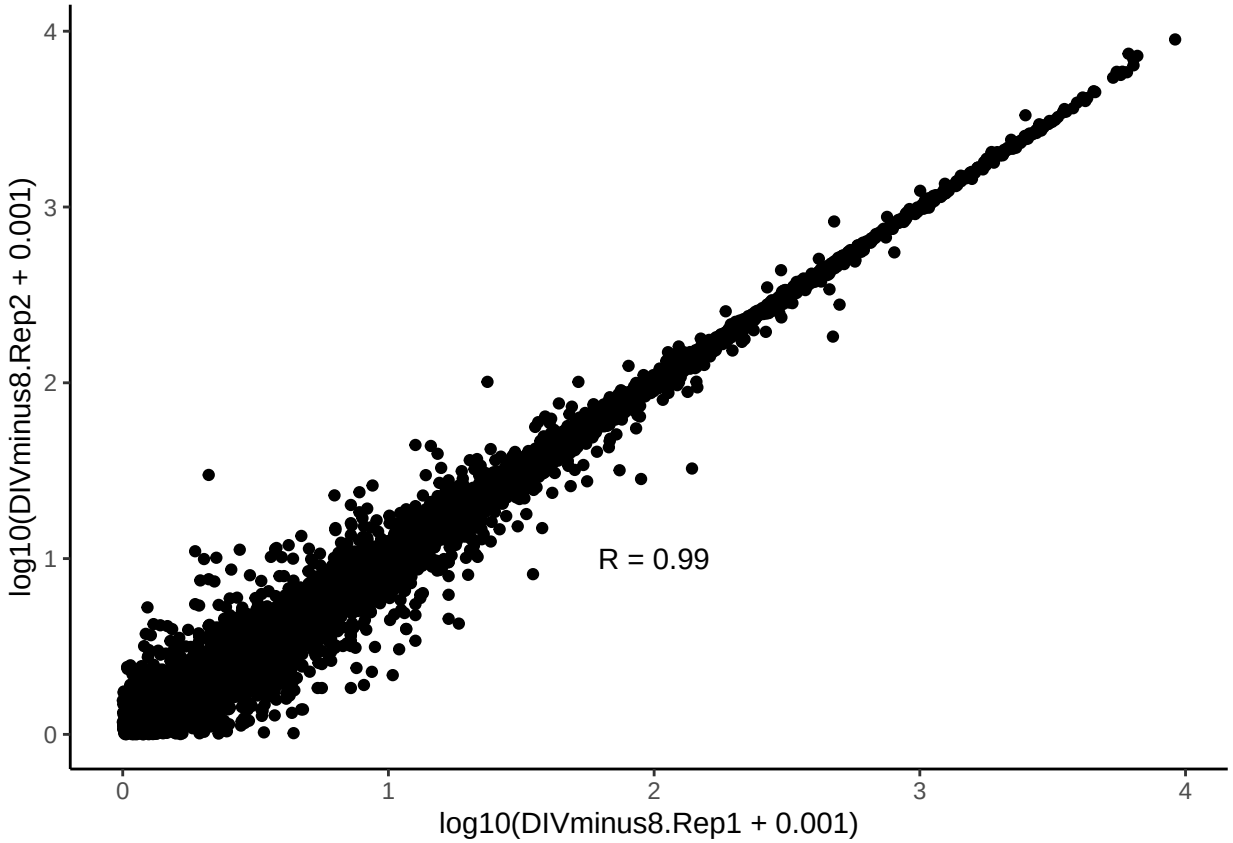
So, for now, we will take another approach of simply using an expression threshold. Any gene that does not meet our threshold will be excluded from the analysis. Obviously where to set this threshold is a bit subjective. For now, we will set this cutoff at 1 TPM.

```
#DIVminus8.Rep1 vs DIVminus8.Rep2

#Since we are plotting log TPM values, we need to add a pseudocount to all samples.
#log(0) is a problem.

#Filter for genes that have expression of at least 1 TPM in both samples
tpms.2samplecor <- dplyr::select(tpms, c(ensembl_gene_id, DIVminus8.Rep1, DIVminus8.Rep2)) %>%
  filter(., DIVminus8.Rep1 >= 1 & DIVminus8.Rep2 >= 1)
```

```
#Add pseudocounts and take log within ggplot function call
r.spearman <- cor.test(tpms.2samplecor$DIVminus8.Rep1, tpms.2samplecor$DIVminus8.Rep2, method = 'spearma
```

```
## Warning in cor.test.default(tpms.2samplecor$DIVminus8.Rep1, tpms.
## 2samplecor$DIVminus8.Rep2, : Cannot compute exact p-value with ties
```

```
r.spearman <- signif(r.spearman, 2)
ggplot(tpms.2samplecor, aes(x = log10(DIVminus8.Rep1 + 1e-3), y = log10(DIVminus8.Rep2 + 1e-3))) +
  geom_point() + theme_classic() +
  annotate('text', x = 2, y = 1, label = paste0('R = ', r.spearman))
```



OK that's two samples compared to each other, but now we want to see how **all** samples compare to **all** other samples. Before we do this we need to decide how to apply our expression cutoff across many samples. Should a gene have to meet the cutoff in only one sample? In all samples? Let's start by saying it has to meet the cutoff in at least half of the 30 samples.

```
#Make a new column in tpms that is the number of samples in which the value is at least 1
tpms.cutoff <- mutate(tpms, nSamples = rowSums(tpms[,2:31] > 1))%>%
  #Now filter for rows where nSamples is at least 15
  #Meaning that at least 15 samples passed the threshold
  filter(., nSamples >= 15) %>%
  #Get rid of the nSamples column
  dplyr::select(., -nSamples)

nrow(tpms)
```

```
## [1] 52346
```

```r
nrow(tpms.cutoff)
```

```
## [1] 15061
```

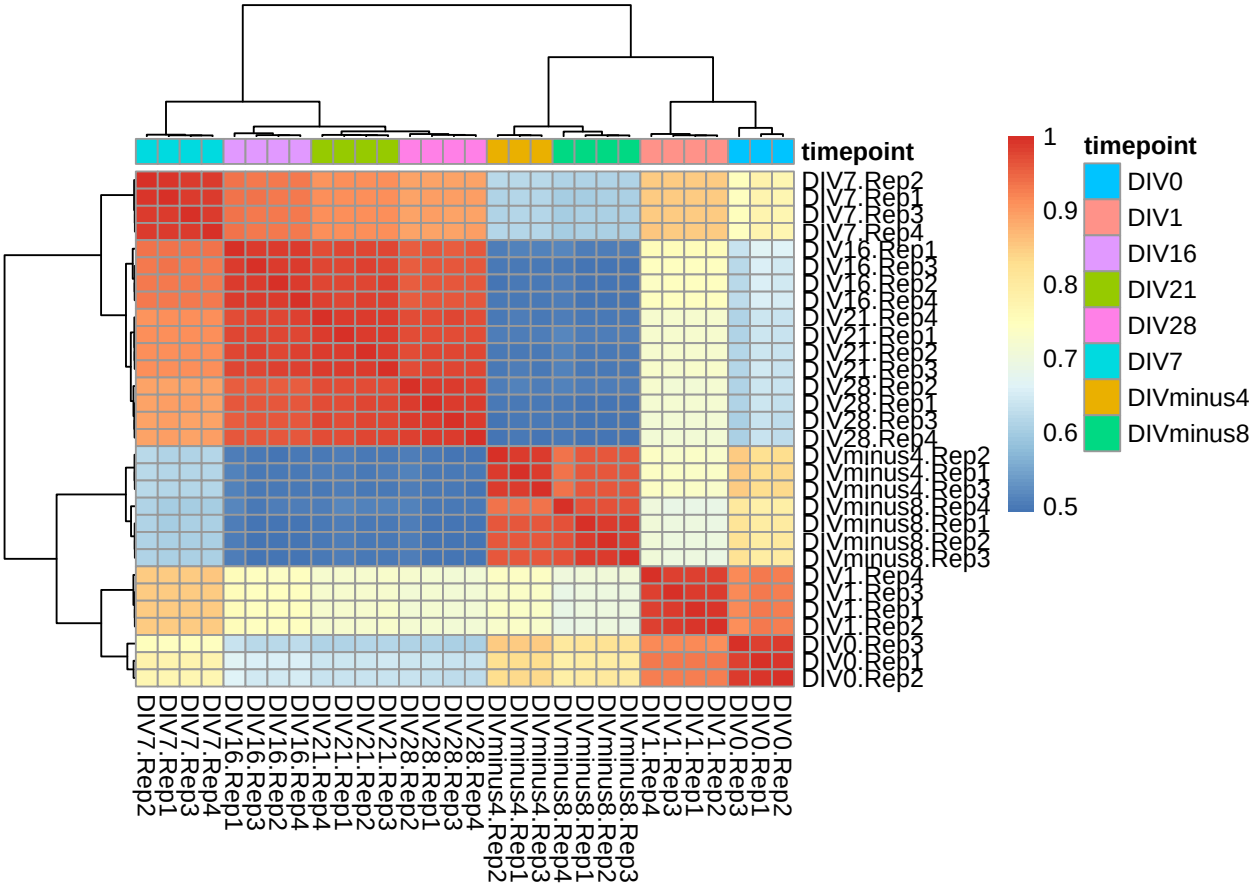
Now we can use the `cor` function to calculate pairwise correlations in a **matrix** of TPM values.

```r
tpms.cutoff.matrix <- dplyr::select(tpms.cutoff, -ensembl_gene_id) %>%
  as.matrix(.)

tpms.cor <- cor(tpms.cutoff.matrix, method = 'spearman')
head(tpms.cor)
```

```
##                 DIVminus8.Rep1 DIVminus8.Rep2 DIVminus8.Rep3 DIVminus8.Rep4
## DIVminus8.Rep1       1.0000000      0.9874291      0.9874863      0.9656629
## DIVminus8.Rep2       0.9874291      1.0000000      0.9887704      0.9663987
## DIVminus8.Rep3       0.9874863      0.9887704      1.0000000      0.9672342
## DIVminus8.Rep4       0.9656629      0.9663987      0.9672342      1.0000000
## DIVminus4.Rep1       0.9613798      0.9618754      0.9611926      0.9386769
## DIVminus4.Rep2       0.9609581      0.9599379      0.9609036      0.9377638
##                 DIVminus4.Rep1 DIVminus4.Rep2 DIVminus4.Rep3 DIV0.Rep1 DIV0.Rep2
## DIVminus8.Rep1       0.9613798      0.9609581      0.9618412 0.7926824 0.7998363
## DIVminus8.Rep2       0.9618754      0.9599379      0.9614563 0.7948024 0.8015281
## DIVminus8.Rep3       0.9611926      0.9609036      0.9613304 0.7943499 0.7994011
## DIVminus8.Rep4       0.9386769      0.9377638      0.9371555 0.7821715 0.7871162
## DIVminus4.Rep1       1.0000000      0.9874579      0.9885148 0.8292617 0.8339424
## DIVminus4.Rep2       0.9874579      1.0000000      0.9873445 0.8254047 0.8313876
##                 DIV0.Rep3 DIV1.Rep1 DIV1.Rep2 DIV1.Rep3 DIV1.Rep4 DIV7.Rep1
## DIVminus8.Rep1 0.8159929 0.6966410 0.6935366 0.6960877 0.7071674 0.6029290
## DIVminus8.Rep2 0.8175660 0.6982362 0.6952722 0.6968994 0.7079007 0.6042110
## DIVminus8.Rep3 0.8171928 0.6985603 0.6942255 0.6964138 0.7070481 0.6043427
## DIVminus8.Rep4 0.8044747 0.6890168 0.6856224 0.6867085 0.6998050 0.6000377
## DIVminus4.Rep1 0.8497808 0.7343311 0.7314782 0.7321168 0.7382880 0.6157548
## DIVminus4.Rep2 0.8480450 0.7329394 0.7289499 0.7298052 0.7357465 0.6113018
##                 DIV7.Rep2 DIV7.Rep3 DIV7.Rep4 DIV16.Rep1 DIV16.Rep2 DIV16.Rep3
## DIVminus8.Rep1 0.6101676 0.6032759 0.6058578  0.5026154  0.4935971  0.4945093
## DIVminus8.Rep2 0.6120526 0.6058035 0.6068240  0.5056898  0.4949182  0.4954424
## DIVminus8.Rep3 0.6125026 0.6038266 0.6059498  0.5050191  0.4942686  0.4936823
## DIVminus8.Rep4 0.6098158 0.6013481 0.6028602  0.5134745  0.5021270  0.4999295
## DIVminus4.Rep1 0.6231006 0.6157895 0.6168821  0.5079881  0.4987724  0.4976697
## DIVminus4.Rep2 0.6198181 0.6123732 0.6135401  0.5052055  0.4958990  0.4962009
##                 DIV16.Rep4 DIV21.Rep1 DIV21.Rep2 DIV21.Rep3 DIV21.Rep4
## DIVminus8.Rep1  0.4942559  0.5015893  0.5011462  0.4978789  0.4978796
## DIVminus8.Rep2  0.4958286  0.5022935  0.5008581  0.4980655  0.4981462
## DIVminus8.Rep3  0.4942954  0.5014684  0.5003943  0.4987814  0.4970675
## DIVminus8.Rep4  0.5021413  0.5114995  0.5091435  0.5087365  0.5042612
## DIVminus4.Rep1  0.4994421  0.5039495  0.5012144  0.5014160  0.5015100
## DIVminus4.Rep2  0.4964639  0.5012931  0.4994238  0.4987792  0.4983438
##                 DIV28.Rep1 DIV28.Rep2 DIV28.Rep3 DIV28.Rep4
## DIVminus8.Rep1  0.4936504  0.5018198  0.4936501  0.4911505
## DIVminus8.Rep2  0.4931310  0.5025217  0.4918232  0.4910483
## DIVminus8.Rep3  0.4926688  0.5028738  0.4917722  0.4917889
## DIVminus8.Rep4  0.4998905  0.5107585  0.4995558  0.4987533
## DIVminus4.Rep1  0.4985176  0.5066587  0.4975275  0.4965792
## DIVminus4.Rep2  0.4975842  0.5040839  0.4958062  0.4952847
```

Now we need to plot these and have similar samples (i.e. those that are highly correlated with each other) be clustered near to each other. We will use `pheatmap` to do this.

```
library(pheatmap)

#Make a dataframe of annotations
annot <- data.frame(row.names = colnames(tpms.cor), timepoint = c(rep('DIVminus8', 4), rep('DIVminus4',
                                                                  rep('DIV0', 3), rep('DIV1', 4), rep('
                                                                  rep('DIV16', 4), rep('DIV21', 4), rep
pheatmap(tpms.cor, annotation_col = annot)
```



This looks pretty good! There are two main points to takeaway here. First, all replicates for a given timepoint are clustering with each other. Second, you can kind of derive the order of the timepoints from the clustering. The biggest separation is between early (DIVminus8 to DIV1) and late (DIV7 to DIV28). After that you can then see finer-grained structure.

PCA analysis

Another way to visualize the relationships and distances between samples is to use a dimensionality reduction technique called Principle Components Anlaysis or PCA. PCA works best when values are approximately normally distributed, so we will first take the log of our expression values.

With our cutoff as it is now (genes have to have expression of at least 1 TPM in half the samples), it is possible that we will have some 0 values. Taking the log of 0 might cause a problem, so we will add a pseudocount.

```
tpms.cutoff.matrix <- dplyr::select(tpms.cutoff, -ensembl_gene_id) %>%
  as.matrix(.)
```

```
#Add pseudocount
tpms.cutoff.matrix <- tpms.cutoff.matrix + 1e-3
#Take log of values
tpms.cutoff.matrix <- log(tpms.cutoff.matrix)
```

OK now we are ready to give this matrix to R's `prcomp` function to find principal components.

```
#prcomp expects samples to be rownames, right now they are columns
#so we need to transpose the matrix using `t`
tpms.pca <- prcomp(t(tpms.cutoff.matrix))

#The coordinates of samples on the principle components are stored in the $x slot
#These are what we are going to use to plot
#We can also also some data about the samples here so that our plot is a little more interesting
tpms.pca.pc <- data.frame(tpms.pca$x) %>%
  mutate(., sample = colnames(tpms.cutoff.matrix)) %>%
  mutate(., timepoint = c(rep('DIVminus8', 4), rep('DIVminus4', 3),
                          rep('DIV0', 3), rep('DIV1', 4), rep('DIV7', 4),
                          rep('DIV16', 4), rep('DIV21', 4), rep('DIV28', 4)))

head(tpms.pca.pc)
```

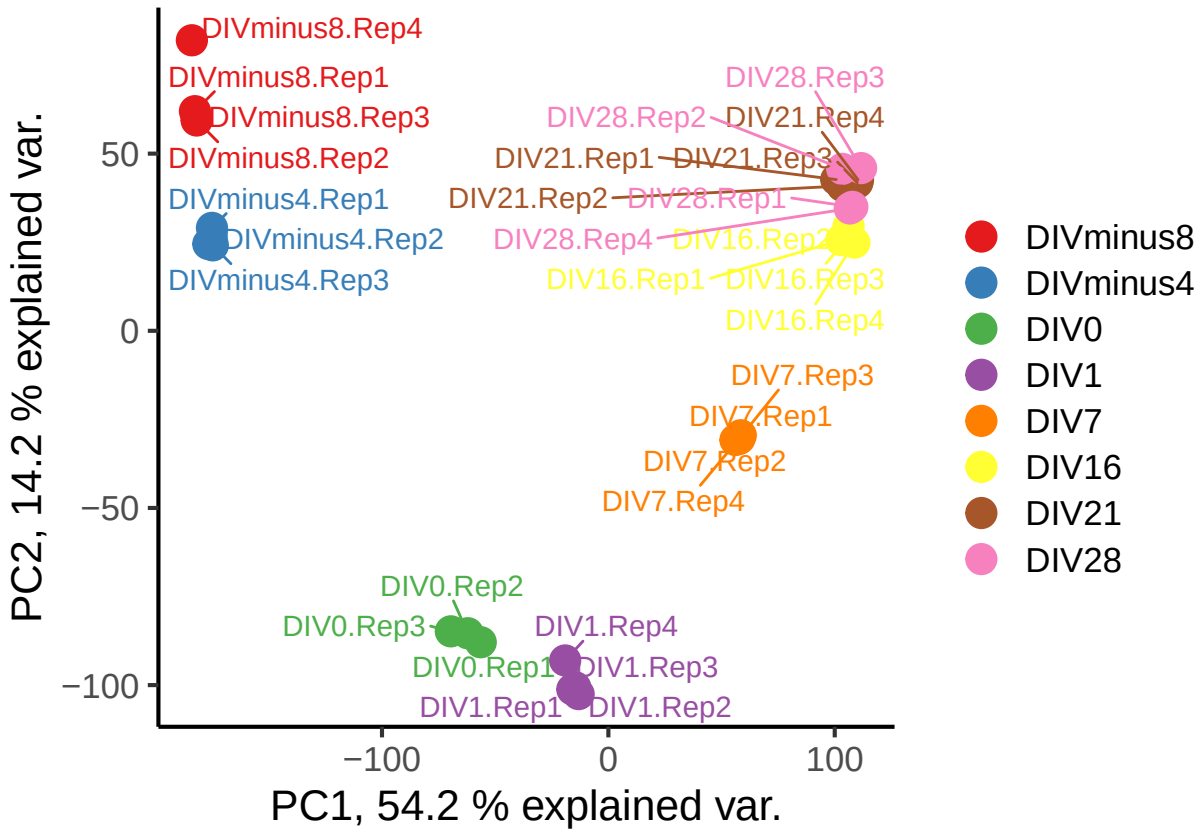
```
##         PC1      PC2       PC3       PC4        PC5       PC6        PC7
## 1 -182.6194 61.98133 -16.17247  23.55376  -1.950990  12.28097 -42.464351
## 2 -181.8714 59.35712 -18.14529  18.21643  -1.863933  19.99230 -22.803068
## 3 -181.8522 60.18284 -16.39260  26.05008   6.091949  21.70283 -16.226412
## 4 -183.9744 81.99765 -15.05918  62.18009  24.116067 -29.56861  48.881938
## 5 -175.1073 29.05140  16.49229 -52.96375 -20.525036 -17.39873  17.522303
## 6 -176.7241 24.51892  18.62796 -52.81598  -1.918574 -21.45545   2.883239
##          PC8        PC9      PC10        PC11       PC12        PC13       PC14
## 1  17.774050 -51.925215  20.72990  23.1132943  -7.184066 -16.3835478  -5.057248
## 2   6.035596  22.957944 -31.23558  -0.1132663  15.505173  23.6769427 -27.500122
## 3  13.898250  12.871829 -15.11476 -29.4121853 -25.471930  17.0694116  29.466639
## 4 -38.315867   6.498666  16.32549   4.2875720  10.516700 -14.8036201  -2.986782
## 5  -5.815204 -23.386192 -23.81985  -7.0821272   6.244212  -0.8584383  33.959272
## 6   5.285993  18.599982  10.38979   0.1731987 -40.156108 -31.5245415 -23.869693
##          PC15       PC16       PC17        PC18       PC19       PC20       PC21
## 1 -11.3051440  6.6775971  20.845329   0.8228503   5.183447  16.465836 -7.232733
## 2  23.8743608  2.3733425 -22.074431  -4.9864550  21.956460  -9.845005  1.390324
## 3   0.8830885 -5.4209496   8.760453   2.2420143 -28.135019  -7.950107  9.878456
## 4 -10.3379320 -0.2818182  -7.950534  -1.7335273  -0.940104  -3.200583  0.453887
## 5 -12.1654459 15.2054570  -6.913991 -28.8152397  25.286453   2.733538 10.320971
## 6  18.3352016 -8.6503056   2.519193  21.0086823  10.654186 -14.053501 -4.593648
##         PC22       PC23        PC24       PC25       PC26       PC27       PC28
## 1  -8.8319524 -0.4009053   3.2118162  0.6500213  0.1146784  1.729793 -3.2469539
## 2 -21.4171947  7.7162545   0.1956315  4.9840353 -4.2234416 11.648665 -1.4556487
## 3  27.4491193 -3.8504270  -1.6409176 -6.9134302  5.1128512 -8.815789  4.0885696
## 4  -0.7347842 -4.1712409  -1.1557132  0.3704460 -1.0501396 -2.453495  0.6760656
## 5   4.2086092  4.4123356 -11.1706642 -3.7323897 -2.0562625 -1.002396  0.5463876
## 6  10.8185478 -4.1312377   5.1110655  3.5388991 -5.8195469  3.236623  1.7144852
##        PC29          PC30         sample timepoint
## 1  5.164827  4.355565e-14 DIVminus8.Rep1 DIVminus8
## 2  2.670914 -5.554715e-14 DIVminus8.Rep2 DIVminus8
## 3 -5.672567  2.912232e-14 DIVminus8.Rep3 DIVminus8
## 4 -1.398131  5.757287e-14 DIVminus8.Rep4 DIVminus8
```

```
## 5  6.840845 -7.103866e-14 DIVminus4.Rep1 DIVminus4
## 6 -4.599421 -1.577558e-14 DIVminus4.Rep2 DIVminus4
```

```
#We can see how much of the total variance is explained by each PC using the summary function
tpms.pca.summary <- summary(tpms.pca)$importance
head(tpms.pca.summary)
```

```
##                             PC1      PC2      PC3      PC4      PC5      PC6
## Standard deviation     115.67194 59.21000 28.73100 24.41998 21.67504 20.44568
## Proportion of Variance   0.54236  0.14211  0.03346  0.02417  0.01904  0.01694
## Cumulative Proportion    0.54236  0.68447  0.71793  0.74210  0.76114  0.77809
##                             PC7      PC8      PC9     PC10     PC11     PC12
## Standard deviation     19.76785 18.74159 18.37137 17.44536 16.99918 16.78242
## Proportion of Variance  0.01584  0.01424  0.01368  0.01234  0.01171  0.01142
## Cumulative Proportion   0.79393  0.80817  0.82185  0.83418  0.84590  0.85731
##                            PC13     PC14     PC15     PC16     PC17     PC18
## Standard deviation     16.43909 16.40473 16.12548 15.72141 15.59795 15.43861
## Proportion of Variance  0.01095  0.01091  0.01054  0.01002  0.00986  0.00966
## Cumulative Proportion   0.86827  0.87918  0.88972  0.89974  0.90960  0.91926
##                            PC19     PC20     PC21     PC22     PC23     PC24
## Standard deviation     15.08524 14.78178 14.70468 14.26197 13.81709 13.24180
## Proportion of Variance  0.00922  0.00886  0.00876  0.00824  0.00774  0.00711
## Cumulative Proportion   0.92848  0.93734  0.94611  0.95435  0.96209  0.96920
##                            PC25     PC26     PC27     PC28     PC29
## Standard deviation     13.05629 12.56717 12.19736 12.05415 11.72298
## Proportion of Variance  0.00691  0.00640  0.00603  0.00589  0.00557
## Cumulative Proportion   0.97611  0.98251  0.98854  0.99443  1.00000
##                               PC30
## Standard deviation     5.136866e-14
## Proportion of Variance 0.000000e+00
## Cumulative Proportion  1.000000e+00
```

```
#The amount of variance explained by PC1 is the second row, first column of this table
#It's given as a fraction of 1, so we multiply it by 100 to get a percentage
pc1var = round(tpms.pca.summary[2,1] * 100, 1)

#The amount of variance explained by PC2 is the second row, second column of this table
pc2var <- round(tpms.pca.summary[2,2] * 100, 1)

#Get decent looking colors.  See RColorBrewer package. This picks 8 colors from the palette Set1
colors <- brewer.pal(8, 'Set1')

#Reorder timepoints explicitly for plotting purposes
tpms.pca.pc$timepoint <- factor(tpms.pca.pc$timepoint, levels = c('DIVminus8', 'DIVminus4', 'DIV0',
                                                                  'DIV1', 'DIV7', 'DIV16', 'DIV21', 'DI

#Plot results
ggplot(tpms.pca.pc, aes(x = PC1, y = PC2, color = timepoint, label = sample)) + geom_point(size = 5)  +
  scale_color_manual(values = colors, name = '') + theme_classic(16) + xlab(paste('PC1,', pc1var, '% ex
  ylab(paste('PC2,', pc2var, '% explained var.')) + geom_text_repel()
```



All done. With this plot you can almost trace the differentiation path. Where is the biggest jump? Which timepoints are very similar?